Krzysztof KAJSTURA [1], Dariusz KANIA [2], Igor KURYTNIK [1]
[1] AKADEMIA TECHNICZNO-HUMANISTYCZNA, KATEDRA ELEKTROTECHNIKI I AUTOMATYKI, ul. Willowa 2, 43-309 Bielsko-Biała
[2] POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI, ul. Akademicka 16, 44-100 Gliwice

Algorytm kodowania stanów wewnętrznych automatu skończonego minimalizujący pobór mocy

Mgr inż. Krzysztof KAJSTURA

Ukończył studia na Wydziale Elektrycznym Politechniki Śląskiej. Jest asystentem w Katedrze Elektrotechniki i Automatyki Akademii Techniczno-Humanistycznej w Bielsku-Białej. Jego zainteresowania naukowe koncentrują się wokół układów cyfrowych i systemów mikroprocesorowych.

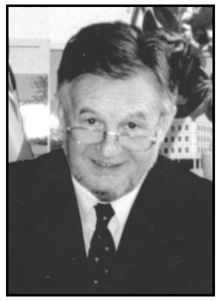
e-mail: kkajstura@ath.bielsko.pl



Prof. dr hab. inż. Igor Piotr KURYTNIK

Profesor zwyczajny ATH w Bielsku-Białej; Kierownik Katedry Elektrotechniki I Automatyki. Z wyróżnieniem ukończył Wydział Automatyki i Elektroniki Politechniki Lwowskiej (1968). Obrona doktoratu 1973, habilitacja 1987; jest autorem ponad 250 patentów i publikacji naukowo-technicznych z zakresu technik informacyjno-pomiarowych. Członek Akademii Inżynierskich w Polsce i Ukrainie.

e-mail: ikurytnik@ath.bielsko.pl



Dr hab. inż. Dariusz KANIA

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 1995, habilitacyjną w 2004r. Jest profesorem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe koncentrują się wokół programowalnych układów i systemów cyfrowych.

e-mail: dkania@polsl.pl



Streszczenie

W artykule przedstawiono nowy algorytm kodowania stanów wewnętrznych automatu skończonego. Głównym zadaniem przedstawionego algorytmu jest minimalizacja poboru mocy w synchronicznych układach sekwencyjnych. Algorytm opiera się na tworzeniu drzewa binarnego, którego węzły powstają na wskutek podziału automatu skończonego. Wysokość drzewa równa jest liczbie bitów słowa kodowego. Wyniki eksperymentów wskazują, że proponowany algorytm prowadzi do zmniejszenia poboru mocy, jak również zmniejszenia powierzchni układu w porównaniu do algorytmów kodowania już opracowanych.

Słowa kluczowe: kodowanie stanów, pobór mocy, automat skończony.

Finite state machine state assignment algorithm for low power dissipation

Abstract

Power consumption has become one of the main issues during the design of embedded systems and VLSI circuits in the recent years, due to the continuous increase in the integration level and the operating frequency. The largest fraction of power consumption in CMOS circuits is caused by signal switches. This paper presents a new algorithm for FSM encoding. The main task of the presented algorithm is to minimise power consumption in synchronous sequential circuits. The algorithm is based on creating a binary tree whose nodes are created by sharing a finite state automaton. The tree height is equal to the number of bits of code words. The algorithm uses the FSM probabilistic model to obtain state encoding that minimise the number of signal transitions. The algorithm has been applied to the MCNC benchmark circuits and has also been compared with other encoding approaches. The experiment results show that the proposed algorithm reduces the power consumption, as well as the circuit area compared to the state encoding algorithms already developed.

Keywords: state assignment, power dissipation, finite state machine.

1. Wprowadzenie

Synteza układów ukierunkowana na redukcję poboru mocy zaczyna obecnie odgrywać coraz większą rolę. Minimalizacja poboru mocy jest szczególnie istotna w układach mobilnych, zasilanych bateryjnie oraz układach pracujących z dużymi częstotliwościami. Nie bez znaczenia jest również ekologiczny aspekt niniejszego zagadnienia. Większość obecnie produkowanych układów cyfrowych wykonywanych jest w technologii CMOS. Pobór mocy w tego typu układach jest sumą statycznego oraz dynamicznego poboru mocy. Pierwszy ze składników związany jest z prądami upływu płynącymi w stanie ustalonym, natomiast drugi z przeładowywaniem pojemności obciążenia w momencie przełączania. Dominującym składnikiem jest drugi z wymienionych, który może być minimalizowany w procesie syntezy logicznej.

Średnia moc pobierana przez układy CMOS jest proporcjonalna do średniej aktywności przełączeń [1]. Prawdopodobieństwo przełączenia i-tego wyjścia p_i definiuje się jako liczbę zmian stanu logicznego na wyjściu i-tej bramki (n_{si}) przypadającą na okres T [2]:

$$p_i = \lim_{T \to \infty} \frac{n_{si}}{T} . \tag{1}$$

Średnia moc dynamiczna wydzielana w układach CMOS, przypadająca na okres T wynosi [3]:

$$P_d = \frac{V_{\mathrm{DD}}^2}{2T} \sum_{i=1}^{n} C_i p_i , \tag{2}$$

gdzie: V_{DD} - napięcie zasilania, C_i - pojemność obciążenia i-tej bramki, n – liczba bramek w układzie.

Istota minimalizacji strat mocy w układach CMOS, uzyskiwana w procesie syntezy logicznej polega na minimalizacji liczby zmian stanów logicznych w układzie. W przypadku sekwencyjnych układów synchronicznych można to osiągnąć poprzez odpowiednie kodowanie stanów wewnętrznych.

Celem artykułu jest przedstawienie oryginalnego algorytmu kodowania stanów wewnętrznych automatów sekwencyjnych ukierunkowanego na minimalizację poboru mocy. Istota zaproponowanego algorytmu sprowadza się do znalezienia takiego skojarzenia słów kodowych z symbolicznie opisanymi stanami, które zapewni podczas występujących w automacie przejść, jak najmniejszą liczbę przełączeń przerzutników.

2. Podstawy teoretyczne, przegląd rozwiązań

Model matematyczny układu sekwencyjnego stanowi automat skończony FSM (ang. *Finite State Machine*). Istotnym elementem syntezy automatów sekwencyjnych jest proces kodowania stanów wewnętrznych. Polega on na przyporządkowaniu każdemu stanowi wewnętrznemu automatu unikatowej reprezentacji binarnej.

Automaty skończone można opisać za pomocą dyskretnych łańcuchów Markowa. Dla podanego prawdopodobieństwa pojawienia się danych stanów logicznych na wejściach automatu, można wyznaczyć prawdopodobieństwo poszczególnych przejść w automacie.

Prawdopodobieństwo statyczne, czyli prawdopodobieństwo znalezienia się automatu w określonym stanie po nieskończenie długim czasie, można obliczyć korzystając z równania Chapmana-Kołmogorowa [4]. Pozwala to na probabilistyczny opis automatu, w którym krawędziom nieskierowanego grafu łączącym odpowiednie stany, przyporządkowuje się wagi poszczególnych przejść.

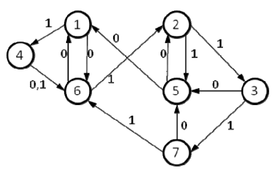
Istota minimalizacji mocy polega na takim kodowaniu, aby stanom połączonym krawędziami o największych wagach, przyporządkować kody, których odległość Hamminga jest równa jeden. Idealnie byłoby wówczas, gdyby wszystkie stany połączone krawędziami miały kody, których odległość Hamminga jest równa jeden. Niestety z uwagi na ograniczenie liczności zbioru kodów, ten warunek z reguły nie może zostać spełniony, szczególnie w przypadku automatów, które posiadają "gęste" grafy. Problem kodowania można zatem sprowadzić do zadania programowania całkowitoliczbowego. Jednak dokładne rozwiązanie tego zadania może być niewykonalne. Dlatego stosuje się metody heurystyczne.

W literaturze przedmiotu znane są różnorodne algorytmy kodowania ukierunkowane na minimalizację poboru mocy. W pracy [3] przedstawiono algorytm POW3 nazywany również kolumnowym. Automat jest kodowany "bit po bicie" przy przestrzeganiu liczności poszczególnych klas nierozróżnialności. W [5] zaproponowano wykorzystanie kodowania Huffmana. Znany jest również algorytm wykorzystujący drzewo rozpinające [6]. Z kolei w pracy [7] zastosowano algorytm wyżarzania. Stosowane są również algorytmy genetyczne [8]. W [9] zaproponowano kilka różnych metod kodowania: Depth First, Min Distance, One Level, One Level Tree, Back Tracking. Metody te składają się z dwóch etapów. Pierwszy polega na odpowiednim sortowaniu stanów, drugi - na przypisywaniu stanom słów kodowych z uwzględnieniem funkcji określającej moc i przydzielone uprzednio kody. W pracy [10] przedstawiono dwa algorytmy: iteracyjny oraz sekwencyjny. Algorytm iteracyjny pozwala na poprawienie wyników kodowania przeprowadzonego za pomocą innego, dowolnego algorytmu. W algorytmie sekwencyjnym kodowanie uzależnione jest od przypisanych uprzednio kodów.

W większości znanych z literatury rozwiązań, proces kodowania stanów odbywa się zgodnie z zasadą "słowo po słowie", w którym kojarzenie słów kodowych z symbolicznymi wartościami odbywa się po uprzednim uporządkowaniu stanów zgodnie z jakimś założonym kryterium. W zaproponowanej metodzie kodowania, nie występuje etap kojarzenia słów kodowych z uporządkowanymi uprzednio, symbolicznie opisanymi stanami. Istota kodowania sprowadza się do podziałów wszystkich stanów na podzbiory, którym przyporządkowuje się kolejno poszczególne bity kodu. Tego typu "dekompozycyjne" kodowanie stanów, w którym kodowanie odbywa się zgodnie z zasadą "bit po bicie", zachowuje elastyczność przyporządkowywania słów kodowych od początku do końca procesu kodowania.

3. Kodowanie stanów wewnętrznych automatu ukierunkowane na minimalizację poboru mocy

Opracowany "dekompozycyjny" algorytm kodowania zostanie przedstawiony na przykładzie kodowania stanów wewnętrznych jednego z automatów testowych MCNC [11]. Wybrano automat dk27, którego graf przedstawiono na rys. 1.



Rys. 1. Graf automatu dk27
Fig. 1. Graph of dk27

Macierz prawdopodobieństwa przejść, wynikająca z sygnałów wejściowych, przy założeniu, że prawdopodobieństwo wystąpienia jedynki i zera na wejściu jest takie samo (p(0)=p(1)=0,5) ma postać:

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 0{,}5 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0{,}5 & 0 & 0{,}5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}5 & 0 & 0{,}5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0{,}5 & 0{,}5 & 0 & 0 & 0 & 0 & 0 \\ 0{,}5 & 0{,}5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}5 & 0{,}5 & 0 \end{bmatrix}. \quad (3)$$

Na postawie powyższej macierzy wyznacza się wektor prawdopodobieństwa statycznego:

$$\mathbf{p}_s = \begin{bmatrix} 0{,}190 & 0{,}190 & 0{,}095 & 0{,}095 & 0{,}167 & 0{,}214 & 0{,}048 \end{bmatrix}. \quad (4)$$

Kolejny krok, to wyznaczenie prawdopodobieństwa przejścia od stanu S_i do stanu S_j pod warunkiem, że automat znajduje się w stanie S_i, zgodnie zależnością:

$$p_t(S_i \to S_j) = p_s(S_i)P(S_{ij}), \quad (5)$$

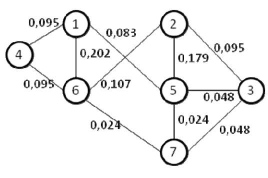
co daje macierz:

$$\mathbf{P}_t = \begin{bmatrix} 0 & 0 & 0 & 0{,}095 & 0 & 0{,}095 & 0 \\ 0 & 0 & 0{,}095 & 0 & 0{,}095 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}048 & 0 & 0{,}048 \\ 0 & 0 & 0 & 0 & 0 & 0{,}095 & 0 \\ 0{,}083 & 0{,}083 & 0 & 0 & 0 & 0 & 0 \\ 0{,}107 & 0{,}107 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}024 & 0{,}024 & 0 \end{bmatrix}. \quad (6)$$

Zmiana stanu automatu z S_i na S_j wymaga przełączenia takiej samej liczby przerzutników, co zmiana ze stanu S_j na S_i. Można zatem przekształcić graf skierowany w graf nieskierowany, którego wagi poszczególnych krawędzi będą przyjmować wartość:

$$w_{ij} = p_t(S_i \to S_j) + p_t(S_j \to S_i). \quad (7)$$

Graf nieskierowany automatu dk27 wraz z wyznaczonymi wagami poszczególnych krawędzi przedstawiono na rys. 2.



Rys. 2. Graf automatu dk27 wraz z wagami krawędzi
Fig. 2. Weighted graph of dk27

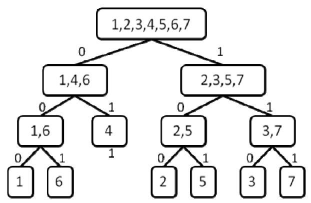
Algorytm kodowania opiera się na tworzeniu drzewa binarnego, którego węzły powstają na wskutek podziału stanów automatu skończonego. Wysokość drzewa równa jest liczbie bitów słowa kodowego. Poszczególne węzły tworzone są następująco:

- jeśli liczba stanów w węźle, który dzielimy jest większa niż dwa, to do nowo tworzonego węzła (lewy syn) są dodawane stany połączone krawędzią o największej wadze, w przeciwnym przypadku są tworzone dwa nowe węzły o liczbie stanów równej jeden,
- jeśli liczba wolnych stanów w dzielonym węźle przekracza maksymalną liczbę stanów (8), to do aktualnego węzła dodawane są stany, których suma wag krawędzi pomiędzy stanami już należącymi do aktualnego węzła jest maksymalna, w przeciwnym przypadku tworzony jest nowy węzeł (prawy syn), w skład którego wchodzą stany, które nie zostały przypisane do poprzednio tworzonego węzła (lewy syn).

Powyższe czynności są powtarzane aż wysokość drzewa binarnego będzie równa liczbie bitów słowa kodowego. Maksymalna liczba stanów w poszczególnych węzłach M_l zależna jest od l – poziomu tworzonego węzła (poziom korzenia wynosi 0) i N – liczby bitów słowa kodowego, zgodnie z zależnością:

$$M_l \leq 2^{N-l}. \tag{8}$$

Obecnie przedstawione zostanie kodowanie automatu testowego dk27. Najpierw, wyznacza się minimalną liczbę bitów potrzebną dla zakodowania siedmiu stanów *N=3*. Następnie tworzy się korzeń drzewa, w skład którego wchodzą wszystkie stany automatu. Na początku wybiera się dwa stany połączone krawędzią o największej wadze, czyli S1 i S6. Po tej operacji liczba wolnych stanów wchodzących w skład korzenia wynosi 5. W przypadku tworzenia węzłów pierwszego poziomu (*l=1*) maksymalna liczba wolnych stanów M_l równa jest 4, zatem należy dodać jeszcze jeden stan do tworzonego węzła drzewa (lewy syn). Wybrany zostaje stan S4, ponieważ suma wag krawędzi łączących ten stan ze stanami S1 i S6 jest maksymalna i wynosi 0,19. Pozostałe stany (S2,S3,S5,S7) zostają dodane do nowego węzła (prawy syn). Kolejne węzły drzewa tworzone są sukcesywnie, aż do momentu, gdy wysokość drzewa będzie równa liczbie bitów słowa kodowego *N*. Binarne drzewo kodowe automatu testowego dk27 przedstawione jest na rys. 3.



Rys. 3. Binarne drzewo kodowania
Fig. 3. Binary tree coding

Zgodnie z otrzymanym drzewem poszczególne stany będą miały następujące kody: s1-0, s2-1, s3-3, s4-6, s5-5, s6-4, s7-7.

4. Wyniki badań eksperymentalnych

Przedstawiony w rozdziale 3 algorytm został zaimplementowany w języku C++. W celu obiektywnego porównania go z innymi metodami znanymi z literatury zaimplementowano również algorytm OLT (One Level Tree) [9]. Wybrano ten algorytm, ponieważ prowadzi do najlepszych rozwiązań pod względem poboru mocy. Do estymacji mocy wykorzystano pakiet SIS [12]. Obliczenia mocy zostały wykonane dla: f=20MHz, V_{DD}=5V. Do minimalizacji części kombinacyjnej układu został wykorzystany skrypt script.rugged [12]. W tabeli 1 przedstawiono wyniki syntezy sześciu wybranych automatów testowych. Poszczególne kolumny przedstawiają pobieraną moc (*P*) oraz powierzchnię zajmowaną przez uzyskiwany układ (*A*). Porównano cztery algorytmy kodowania: algorytm "one-hot" (gorąca jedynka), algorytm zaimplementowany w module Yedi, znany z literatury algorytm OLT [9] oraz przestawiony w niniejszej publikacji algorytm dekompozycyjny (AD). W ostatnim wierszu tabeli 1 zawarto liczby określające sumaryczną moc i całkowitą powierzchnię.

Tab. 1. Wyniki badań eksperymentalnych, P-moc, A-powierzchnia
Tab. 1. Experimental results, P-power, A-area

FSM	One hot		Yedi		OLT		AD	
	P μW	A	P μW	A	P μW	A	P μW	A
cse	356	462	389	307	381	311	350	296
dk15	395	139	373	111	387	119	373	111
dk27	262	114	233	60	175	65	168	56
ex1	571	526	720	366	564	370	548	376
lion9	310	146	121	60	183	73	110	69
train4	169	65	88	36	80	36	84	36
suma	**2063**	**1452**	**1924**	**940**	**1770**	**974**	**1633**	**944**

W przypadku pięciu automatów zaproponowany algorytm (AD) zapewnił uzyskanie najlepszych wyników. W porównaniu do popularnego i powszechnie stosowanego algorytmu "one hot", algorytm AD w każdym z przypadków doprowadził do redukcji poboru mocy. Dla automatu lion9 uzyskano zmniejszenie mocy aż o 64%. Kodując stany algorytmem Yedi, tylko dla automatu dk15 osiągnięto moc identyczną jak w proponowanej metodzie AD. Maksymalna różnica pomiędzy YEDI i AD pojawiła się dla automatu dk27 i wynosiła 39%. Proponowany algorytm kodowania doprowadził również do lepszych wyników od algorytmu OLT. Sumaryczna moc dla wszystkich automatów była najmniejsza dla algorytmu AD.

5. Podsumowanie

W artykule przedstawiono nowy, dekompozycyjny algorytm kodowania stanów wewnętrznych automatu skończonego. Wyniki eksperymentów wskazują jednoznacznie, że prowadzi on do zmniejszenia poboru mocy w stosunku do popularnego algorytmu "one hot", ukierunkowanego na minimalizację powierzchni, bardzo wyrafinowanego i skutecznego algorytmu Yedi oraz znanego z bardzo efektywnych pod względem mocy algorytmu OLT. Oczywiście, w celu bardziej obiektywnego porównania algorytmów kodowania konieczne jest przeprowadzenie znacznie szerszego wachlarza eksperymentów. W pierwszej kolejności zostaną jednak sprawdzone różne warianty kojarzenia wierzchołków drzewa binarnego z kolejnymi bitami słów kodowych, ponieważ widać duży wpływ tego procesu na ostateczne wyniki. Można więc liczyć, na uzyskanie jeszcze lepszych rezultatów.

Podsumowując, warto zwrócić uwagę na dwie, bardzo korzystne cechy przestawionego algorytmu kodowania. Po pierwsze, jest on niezwykle "szybkim" algorytmem. Dodatkowo prowadzi do efektywnych rozwiązań pod względem powierzchni. W porównaniu do nastawionego na minimalizację powierzchni algorytmu Yedi, uzyskano sumaryczną redukcję mocy o 15,1%, przy wzroście powierzchni o 0,4%.

6. Literatura

[1] Cirit M.A.: Estimating dynamic power consumption of CMOS circuits. Proc. IEEE Int. Conf. CAD, Nov. 1987, pp. 534-537.
[2] Kentzer K., Ghosh A., Devadas S., White J.: Estimation of average switching activity in combinational and sequential circuits. Proc. Design Automation Conf., June 1992, pp. 253-259.
[3] Benini L., DeMicheli G.: State Assignment for Low Power Dissipation. In IEEE Journal on Solid-state Circuits, Vol. 30, No. 3, 1995.
[4] Freitas A. T., Oliveira A. L.: Implicit Resolution of the Chapman-Kolmogorov Equations for Sequential Circuits: An Application in Power Estimation. Proceedings of DATE, 2003.
[5] Surti P., Chao L.F.: Lower Power FSM Design Using Huffman-Style Encoding. IEEE EDTC-97, 1997, pp. 521-525.
[6] Nőth W., Kolla R.: Spanning Tree Based State Encoding for Lower Power Dissipation. Technical Report, University of Wűrzburg, 1998.
[7] Roy K., Prasad S. C.: Circuit Activity Based Logic Synthesis for Low Power Reliable Operations. Trans. on VLSI Systems, Vol. 1, No. 4, 1993.
[8] Venkataraman G., Reddy S.M., Pomeranz I.: GALLOP: Genetic Algorithm Based Lowe Power FSM Synthesis by Simultaneous Partitioning and State Assignment. 6th Intl. on VLSI Design, 2003.
[9] Baccheta P., L. Daldoss, D. Sciuto, C. Silvano: Lower-Power State Assignment Techniques for Finite State Machines. IEEE International Symposium on Circuits and Systems (ISCAS'00), 2000.
[10] Salauyou V., Grześ T.: Badania algorytmów kodowania stanów wewnętrznych automatu skończonego zorientowanych na minimalizację poboru mocy. Pomiary, Automatyka, Kontrola, 2008, R.54, nr 8.
[11] Yang S.: Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0. Technical Report, Microelectronics Center of North Carolina, 1991.
[12] Sentovich, E. M., Singh, K. J., Lavango, L., Moon, C., Muragi, R., Saldhana, A., Savoj, H., Stephen, P., Brayton, R., Sangiovanni-Vincentelli A.: SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, University of California, Berkeley, 1992.

otrzymano / received: 24.05.2010
przyjęto do druku / accepted: 02.07.2010 *artykuł recenzowany*